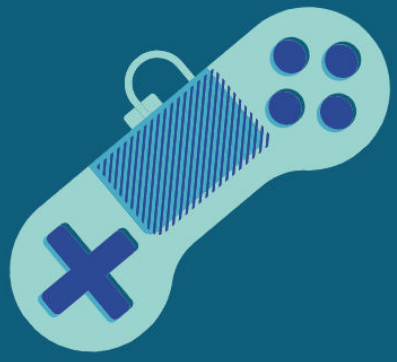
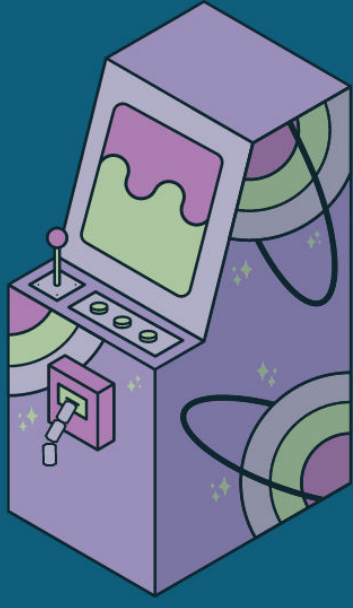
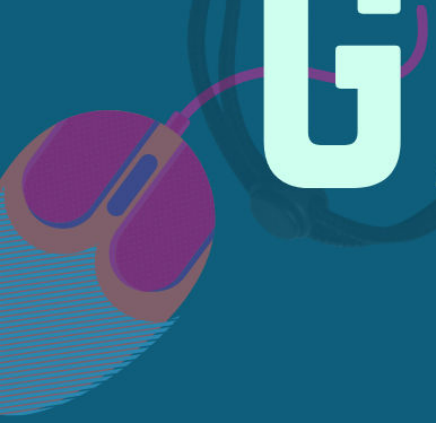




Symbiosis Institute of Technology

PYTHON MINI PROJECT REPORT

GALAGA



MADE BY-
SAKSHAM GUPTA (21070122137)
SAAHIL SHAIKH (21070122132)
SACHIT DESAI (21070122134)

OBJECTIVE:

The objective of our program is to create a game wherein the user can:

1. Control the hero to move left and right
2. Shoot lasers to eliminate constantly moving enemies through a field of obstacles
3. Increment the score depending on the type of enemy eliminated
4. Lose lives for each direct hit to the player
5. Win the game when each enemy has been eliminated
6. Lose the game when all lives have been lost

This is to be achieved by using previously learned topics such as variables, functions, objects, classes, and modules.

MOTIVATION:

Sharing a common interest in video games from the early 80's, the idea of making our own retro-style video game came quite easily to the three of us. Companies like SEGA, Nintendo, NAMCO, and Atari continued to make ground-breaking games throughout the late 70's and early 80's and laid down the foundations of the video game industry and the gaming culture we see today. Going from one day believing that it would be impossible to play a game on a portable device, to the release of the revolutionary 'GameBoy' in 1989, it definitely inspired many tech enthusiasts to not only develop the game industry, but also to develop technology as a whole.

Many of us have heard stories about how Steve Wozniak and other future co-founders of 'Atari' made the hit game 'breakout' using only 'BASIC'. Making a video game with such a rudimentary programming language was definitely a big feat at the time, and the availability of advanced languages such as python and Java have allowed us to develop similar games much faster and much more efficiently today. Wozniak’s story is one of many reasons as to why we were motivated to make a video game, and will complete our objective with the help of the Pygame library of Python.

INTRODUCTION TO PROJECT:

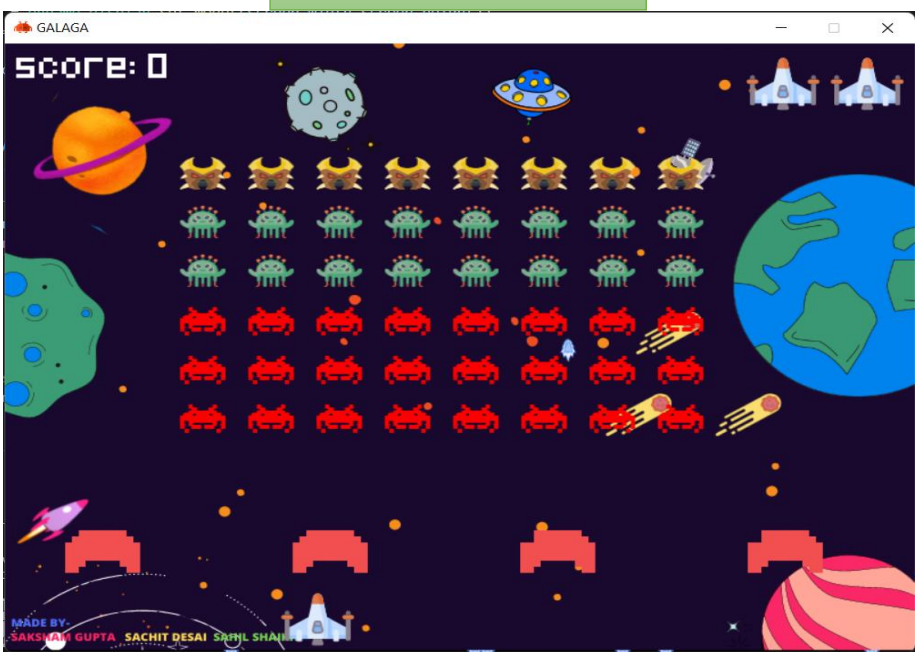
Galaga was a very popular video game in 80s and it created a revolution in the world of Arcade Gaming. Almost every millennial is aware about this game. We have given some modern touch to it, keeping the roots same. Our team has tried to recreate the magic of the video game using Pygame Library.

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. Pygame was originally written by Pete Shinners to replace PySDL after its development stalled. It has been a community project since 2000 and is released under the free software GNU Lesser General Public License (which "provides for Pygame to be distributed with open source and commercial software").
Example of few popular games are as follows:

- Frets on Fire.
- Dangerous High School Girls in Trouble.
- Save the Date, IndieCade 2013 Finalist.

ORIGINAL GALAGA

OUR GALAGA



PROGRAM:

The overall program consists of the following modules :-

- **main.py**
- **player.py**
- **laser.py**
- **obstacle.py**
- **alien.py**

main.py

```python
import pygame, sys
from player import Player
import obstacle
from alien import Alien, Extra
from random import choice, randint
from laser import Laser

class Game:
    def __init__(self):
        # Player setup
        player_sprite = Player((screen_width / 2,screen_height),screen_width,5)
        self.player = pygame.sprite.GroupSingle(player_sprite)

        # health and score setup
        self.lives =3
        self.live_surf = pygame.image.load('../graphics/player.png').convert_alpha()
        self.live_x_start_pos = screen_width - (self.live_surf.get_size()[0] * 2 + 20)
        self.score = 0
        self.font = pygame.font.Font('../font/Pixeled.ttf',20)

        # Obstacle setup
        self.shape = obstacle.shape
        self.block_size = 6
        self.blocks = pygame.sprite.Group()
        self.obstacle_amount = 4
        self.obstacle_x_positions = [num * (screen_width / self.obstacle_amount) for num in range(self.obstacle_amount)]
        self.create_multiple_obstacles(*self.obstacle_x_positions, x_start = screen_width / 15, y_start = 480)

        # Alien setup
        self.aliens = pygame.sprite.Group()
        self.alien_lasers = pygame.sprite.Group()
        self.alien_setup(rows = 6, cols = 8)
        self.alien_direction = 1

        # Extra setup
        self.extra = pygame.sprite.GroupSingle()
        self.extra_spawn_time = randint(40,80)

        # Audio
        music = pygame.mixer.Sound('../audio/music.wav')
        music.set_volume(0.2)
        music.play(loops = -1)
        self.laser_sound = pygame.mixer.Sound('../audio/laser.wav')
        self.laser_sound.set_volume(0.5)
        self.explosion_sound = pygame.mixer.Sound('../audio/explosion.wav')
        self.explosion_sound.set_volume(0.3)

    def create_obstacle(self, x_start, y_start,offset_x):
        for row_index, row in enumerate(self.shape):
            for col_index,col in enumerate(row):
                if col == 'x':
                    x = x_start + col_index * self.block_size + offset_x
                    y = y_start + row_index * self.block_size
                    block = obstacle.Block(self.block_size,(241,79,80),x,y)
                    self.blocks.add(block)

    def create_multiple_obstacles(self,*offset,x_start,y_start):
        for offset_x in offset:
            self.create_obstacle(x_start,y_start,offset_x)

    def alien_setup(self,rows,cols,x_distance = 60,y_distance = 48,x_offset = 70, y_offset = 100):
        for row_index, row in enumerate(range(rows)):
            for col_index, col in enumerate(range(cols)):
                x = col_index * x_distance + x_offset
                y = row_index * y_distance + y_offset

                if row_index == 0: alien_sprite = Alien('yellow',x,y)
                elif 1 <= row_index <= 2: alien_sprite = Alien('green',x,y)
                else: alien_sprite = Alien('red',x,y)
                self.aliens.add(alien_sprite)

    def alien_position_checker(self):
        all_aliens = self.aliens.sprites()
        for alien in all_aliens:
            if alien.rect.right >= screen_width:
                self.alien_direction = -1
                self.alien_move_down(2)
            elif alien.rect.left <= 0:
                self.alien_direction = 1
                self.alien_move_down(2)

    def alien_move_down(self,distance):
        if self.aliens:
            for alien in self.aliens.sprites():
                alien.rect.y += distance

    def alien_shoot(self):
        if self.aliens.sprites():
            random_alien = choice(self.aliens.sprites())
            laser_sprite = Laser(random_alien.rect.center,6,screen_height)
            self.alien_lasers.add(laser_sprite)
            self.laser_sound.play()

    def extra_alien_timer(self):
        self.extra_spawn_time -= 1
        if self.extra_spawn_time <= 0:
            self.extra.add(Extra(choice(['right','left']),screen_width))
            self.extra_spawn_time = randint(400,800)

    def collision_checks(self):

        # player lasers
        if self.player.sprite.lasers:
            for laser in self.player.sprite.lasers:
                # obstacle collisions
                if pygame.sprite.spritecollide(laser,self.blocks,True):
                    laser.kill()


                # alien collisions
                aliens_hit = pygame.sprite.spritecollide(laser,self.aliens,True)
                if aliens_hit:
                    for alien in aliens_hit:
                        self.score += alien.value
                    laser.kill()
                    self.explosion_sound.play()

                # extra collision
                if pygame.sprite.spritecollide(laser,self.extra,True):
                    self.score += 500
                    laser.kill()

        # alien lasers
        if self.alien_lasers:
            for laser in self.alien_lasers:
                # obstacle collisions
                if pygame.sprite.spritecollide(laser,self.blocks,True):
                    laser.kill()

                if pygame.sprite.spritecollide(laser,self.player,False):
                    laser.kill()
                    self.lives -= 1
                    if self.lives <= 0:
                        pygame.quit()
                        sys.exit()

        # aliens
        if self.aliens:
            for alien in self.aliens:
                pygame.sprite.spritecollide(alien,self.blocks,True)

                if pygame.sprite.spritecollide(alien,self.player,False):
                    pygame.quit()
                    sys.exit()

    def display_lives(self):
        for live in range(self.lives - 1):
            x = self.live_x_start_pos + (live * (self.live_surf.get_size()[0] + 10))
            screen.blit(self.live_surf,(x,8))

    def display_score(self):
        score_surf = self.font.render(f'score: {self.score}',False,'white')
        score_rect = score_surf.get_rect(topleft = (10,-10))
        screen.blit(score_surf,score_rect)

    def victory_message(self):
        if not self.aliens.sprites():
            victory_surf = self.font.render('VICTORY',False,'white')
            victory_rect = victory_surf.get_rect(center = (screen_width / 2, screen_height / 2))
            screen.blit(victory_surf,victory_rect)

    def run(self):
        self.player.update()
        self.alien_lasers.update()
        self.extra.update()

        self.aliens.update(self.alien_direction)
        self.alien_position_checker()
        self.extra_alien_timer()
        self.collision_checks()

        self.player.sprite.lasers.draw(screen)
        self.player.draw(screen)
        self.blocks.draw(screen)
        self.aliens.draw(screen)
        self.alien_lasers.draw(screen)
        self.extra.draw(screen)
        self.display_lives()
        self.display_score()
        self.victory_message()


if __name__ == '__main__':
    pygame.init()
    screen_width = 800
    screen_height = 600

    pygame.display.set_caption("GALAGA")
    pygame.display.set_icon(pygame.image.load('../graphics/icon.png'))

    screen = pygame.display.set_mode((screen_width,screen_height))
    clock = pygame.time.Clock()

    background = pygame.image.load('../graphics/background.png')
    game = Game()


    ALIENLASER = pygame.USEREVENT + 1
    pygame.time.set_timer(ALIENLASER,800)

    while True:
        screen.blit(background, (0, 0))
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == ALIENLASER:
                game.alien_shoot()

        # screen.fill((30,30,30))
        game.run()


        pygame.display.flip()
        clock.tick(60)
```

player.py

```python
import pygame
from laser import Laser

# Making Player Class
class Player(pygame.sprite.Sprite):
    def __init__(self,pos,constraint,speed):
        super().__init__()
        self.image = pygame.image.load('../graphics/player.png').convert_alpha()
        self.rect = self.image.get_rect(midbottom = pos)
        self.speed = speed
        self.max_x_constraint = constraint
        self.ready = True
        self.laser_time = 0
        self.laser_cooldown = 600

        self.lasers = pygame.sprite.Group()

        self.laser_sound = pygame.mixer.Sound('../audio/laser.wav')
        self.laser_sound.set_volume(0.5)

    def get_input(self):
        keys = pygame.key.get_pressed()

        if keys[pygame.K_RIGHT] or keys[pygame.K_d]:
            self.rect.x += self.speed
        elif keys[pygame.K_LEFT] or keys[pygame.K_a]:
            self.rect.x -= self.speed

        if keys[pygame.K_SPACE] and self.ready:
            self.shoot_laser()
            self.ready = False
            self.laser_time = pygame.time.get_ticks()
            self.laser_sound.play()

    def recharge(self):
        if not self.ready:
            current_time = pygame.time.get_ticks()
            if current_time - self.laser_time >= self.laser_cooldown:
                self.ready = True

    def constraint(self):
        if self.rect.left <= 0:
            self.rect.left = 0
        if self.rect.right >= self.max_x_constraint:
            self.rect.right = self.max_x_constraint

    def shoot_laser(self):
        self.lasers.add(Laser(self.rect.center,-8,self.rect.bottom))

    def update(self):
        self.get_input()
        self.constraint()
        self.recharge()
        self.lasers.update()
```

laser.py

```python
import pygame

class Laser(pygame.sprite.Sprite):
    def __init__(self,pos,speed,screen_height):
        super().__init__()
        self.image = pygame.image.load('../graphics/bullet.png').convert_alpha()
        self.rect=self.image.get_rect(center=pos)
        self.speed = speed
        self.height_y_constraint = screen_height


    def destroy(self):
        if self.rect.y <= -50 or self.rect.y >= self.height_y_constraint + 50:
            self.kill()

    def update(self):
        self.rect.y += self.speed
        self.destroy()

```

obstacle.py

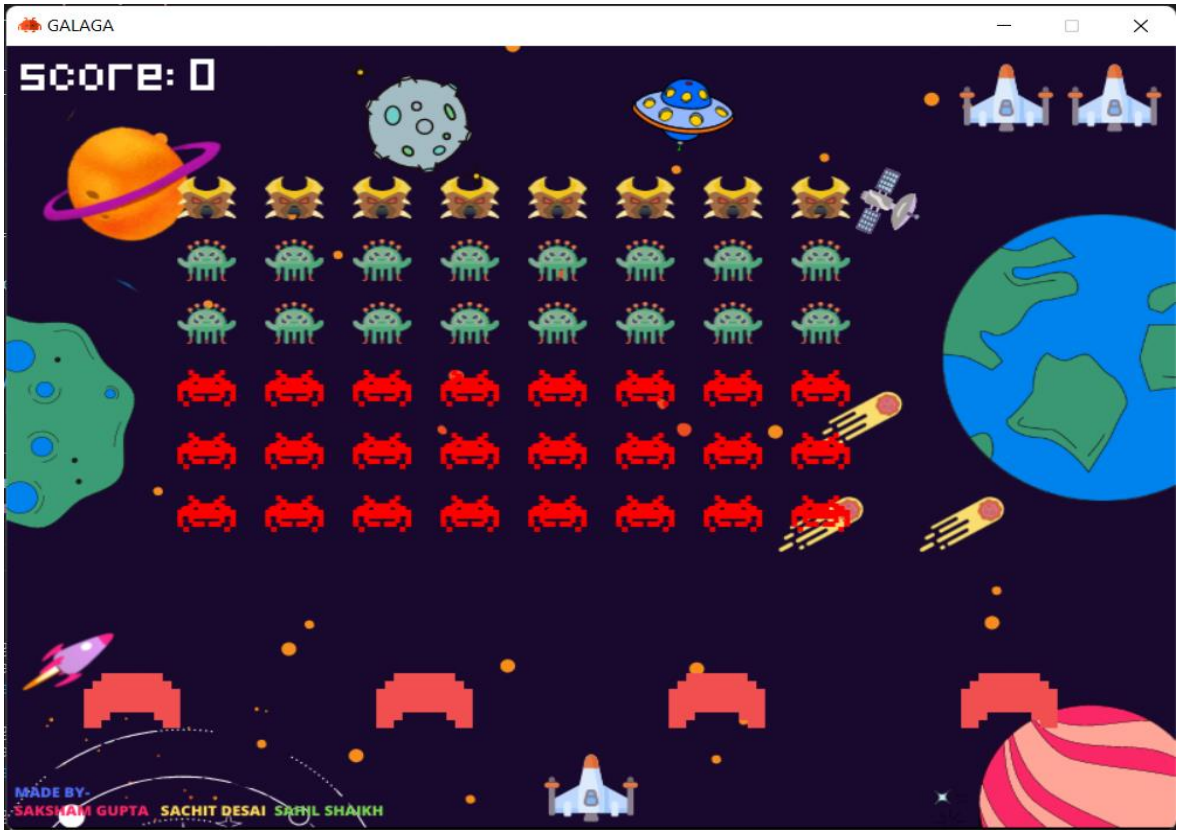
```python
import pygame

class Block(pygame.sprite.Sprite):
    def __init__(self,size,color,x,y):
        super().__init__()
        self.image = pygame.Surface((size,size))
        self.image.fill(color)
        self.rect = self.image.get_rect(topleft = (x,y))

shape = [
'  xxxxxxx',
' xxxxxxxxx',
'xxxxxxxxxxx',
'xxxxxxxxxxx',
'xxxxxxxxxxx',
'xxx     xxx',
'xx       xx']
```

alien.py

```python
import pygame

class Alien(pygame.sprite.Sprite):
    def __init__(self,color,x,y):
        super().__init__()
        file_path = '../graphics/' + color + '.png'
        self.image = pygame.image.load(file_path).convert_alpha()
        self.rect = self.image.get_rect(topleft = (x,y))

        if color == 'red': self.value = 100
        elif color == 'green': self.value = 200
        else: self.value = 300

    def update(self,direction):
        self.rect.x += direction

class Extra(pygame.sprite.Sprite):
    def __init__(self,side,screen_width):
        super().__init__()
        self.image = pygame.image.load('../graphics/extra.png').convert_alpha()

        if side == 'right':
            x = screen_width + 50
            self.speed = - 3
        else:
            x = -50
            self.speed = 3

        self.rect = self.image.get_rect(topleft = (x,80))

    def update(self):
        self.rect.x += self.speed
```

RESULT AND DISCUSSION:

After running the Program, the user will be able to play the game on his laptop with the help of his keyboard and the basic controls of WASD, arrow keys, and the space bar. The game will continue till the enemies get eliminated or the player runs out of the allotted three lives for the game. A victory bar is displayed once the player wins the game. The group members learned the fundamentals of Pygame and understood the concept of Classes and Objects. As far as the entire project is concerned, it was not an easy task to learn and implement the fundamentals on this scale and helped us get out of our comfort zone and discover more about the language of Python.



CONCLUSION:

Our team has worked hard to create this beautiful game from the era which paved the way for many modern games. The game brings back a wave of nostalgia from when arcades were booming, with a modern design. In an era where gaming has gone to levels where we are almost close to feeling the game's environment with the help of modern technologies, we must not forget to appreciate the things that are the reason this industry is what it is today. And as an ode to the classics, we present 'Galaga' to you.

REFERENCES:

freeCodeCamp.org

Pygame Tutorial for Beginners - Python Game Development Course - YouTube

www.pygame.org